Spring 5-1-1989

# Tensor Methods for Unconstrained Optimization ; CU-CS-439-89

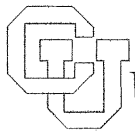
Robert B. Schnabel
*University of Colorado Boulder*

Ta-Tung Chow
*University of Colorado Boulder*

# Tensor Methods for Unconstrained Optimization Using Second Derivatives *

**Robert B. Schnabel**
**Ta-Tung Chow**

**CU-CS-439-89**



**University of Colorado at Boulder**
**DEPARTMENT OF COMPUTER SCIENCE**

* This research was supported by ARO grant DAAL 03-88-K-0086 and NSF grant CCR-8702403.

## Abstract

We introduce a new type of method for unconstrained optimization, which we call a tensor method. It is related in its basic philosophy to the tensor methods for nonlinear equations of Schnabel and Frank, but beyond that the methods have significant differences. The tensor method for unconstrained optimization bases each iteration upon a fourth order model of the objective function. This model consists of the quadratic portion of the Taylor series, plus low rank third and fourth order terms that cause the model to interpolate already calculated function and gradient values from one or more previous iterates. We show that the costs of forming, storing, and solving the tensor model are not significantly more than these costs for a standard method based upon a quadratic Taylor series model. Test results are presented for sets of problems where the Hessian at the minimizer is nonsingular, and where it is singular. On all the test sets, the tensor method solves considerably more problems than a comparable standard method. On problems solved by both methods, the tensor method requires about half as many iterations, and half as many function and derivative evaluations as the standard method, on the average.

## 1. Introduction

This paper describes a new method, called a *tensor method*, for solving the unconstrained optimization problem

$$\text{given } f : R^n \to R \ , \ \text{find } x_* \in R^n \text{ such that } f(x_*) \leq f(x) \text{ for all } x \in D \tag{1.1}$$

where $D$ is some open neighborhood containing $x_*$. We assume that $f(x)$ is at least twice continuously differentiable, and that $n$ is of moderate size, say $n < 100$. Our objective is to create a general purpose method that is more reliable and efficient than state-of-the-art methods for solving such problems, particularly in cases where the evaluation of $f(x)$ and its derivatives is expensive. We especially intend to improve upon the efficiency and reliability of standard methods on problems where $\nabla^2 f(x_*)$ is singular.

The distinguishing feature of our new method is that it bases each iteration upon a fourth order model of $f(x)$, as opposed to the standard quadratic model. The third and fourth order terms of this model have special, low-rank forms that make the costs of using the higher order model reasonable. In particular, in comparison to standard methods, the formation and use of the fourth order tensor model requires no additional function or derivative evaluations per iteration, only a small number of additional arithmetic operations per iteration, and only a very small amount of additional storage.

The tensor method approach was introduced by Schnabel and Frank [1984, 1987], who describe tensor methods for solving systems of nonlinear equations. Their methods base each iteration of an algorithm for solving $F(x) = 0$, where $F : R^n \to R^n$, upon the second order model

$$M_T(x_c + d) = F(x_c) + J_c \, d + \tfrac{1}{2} T_c \, d \, d \ . \tag{1.2}$$

Here $x_c$ is the current iterate, $J_c \in R^{n \times n}$ is the Jacobian matrix $F'(x_c)$ or an approximation to it, and $T_c \in R^{n \times n \times n}$ is a low rank "tensor". In Schnabel and Frank's computational experiments, the use of the tensor methods led to significant improvements in efficiency and reliability over state-of-the-art methods for nonlinear equations that are based upon standard, linear models. In the case when $F'(x_c)$ is available,

the average reductions measured in function and derivative evaluations ranged from 20% to 60%, on both nonsingular and singular problems. Frank [1984] also proved that this derivative tensor method has a three-step, order 1.16 local convergence rate on problems where rank($F'(x_*)$) = $n-1$, whereas standard methods are linearly convergent under these conditions.

The tensor method described in this paper is related to the methods of Schnabel and Frank in its basic philosophy, but it is not a straightforward generalization of their methods. In particular, it is *not* the application of the model (1.2) to the problem $\nabla f(x) = 0$. This would correspond to using a third order model of $f(x)$; as we have already stated, we use a fourth order model instead. To help motivate the basic differences, we first summarize some features of standard methods for unconstrained optimization.

Standard methods for solving small to moderate size unconstrained optimization problems base each iteration upon a quadratic model of $f(x)$ around the current iterate $x_c$,

$$m(x_c+d) = f(x_c) + g_c^T d + \tfrac{1}{2} d^T H_c d \ , \tag{1.3}$$

where $d \in R^n$, $g_c \in R^n$ is $\nabla f(x_c)$ or a finite difference approximation to it, and $H_c \in R^{n \times n}$. Such methods can be divided into two classes, those where $H_c$ is $\nabla^2 f(x_c)$ or a finite difference approximation to it, and those where $H_c$ is a secant approximation to the Hessian formed solely from current and previous gradient values. In this paper we will consider standard and tensor methods of the first type, where both $\nabla f(x_c)$ and $\nabla^2 f(x_c)$ are available analytically or by finite differences at each iteration. A subsequent paper will discuss tensor methods for unconstrained optimization that are based solely on function and gradient values.

The fundamental method for unconstrained optimization, Newton's method, is defined when $\nabla^2 f(x_c)$ is nonsingular. It consists of using $H_c = \nabla^2 f(x_c)$ in (1.3) and setting the next iterate $x_+$ to the critical point of (1.3),

$$x_+ = x_c - \nabla^2 f(x_c)^{-1} \nabla f(x_c) \ . \tag{1.4}$$

The basic properties of Newton's method are well known. If $\nabla^2 f(x_*)$ is nonsingular at a local minimizer

$x_*$, and the initial iterate is sufficiently close to $x_*$, then the sequence of iterative generated by (1.4) converges quadratically to $x_*$. If the initial iterate is not sufficiently close to $x_*$, then the iterates produced by Newton's method may not converge to $x_*$, but they may be made to converge through the use of line search or trust region strategies (see e.g. Fletcher [1980], Gill, Murray, and Wright [1981], Dennis and Schnabel [1983]). The main costs of unconstrained optimization methods based upon Newton's method are : one evaluation of $\nabla^2 f(x)$, and one or more evaluations of $\nabla f(x)$ and $f(x)$, at each iteration; the solution of a symmetric $n \times n$ system of linear equations at each iteration, costing a small multiple of $n^3$ arithmetic operations; and the storage of a symmetric $n \times n$ matrix.

One shortcoming of standard unconstrained minimization methods is that they do not converge quickly if the Hessian at the minimizer, $\nabla^2 f(x_*)$, is singular. Griewank and Osborne [1983] have shown that in this case, the iterates produced by (1.4) generally are at best linearly convergent, even if $\nabla^2 f(x_c)$ is non-singular at all the iterates. Furthermore, the third derivatives do not supply information in the direction(s) where the second derivative matrix is lacking, since the necessary conditions for minimization show that at any minimizer $x_*$ where $\nabla^2 f(x_*)$ is singular with null vector $v$, $\nabla^3 f(x_*)vvd$ must also be 0 for all $d \in R^n$. Thus, adding an approximation to $\nabla^3 f(x_c)$ alone will not lead to better than linear convergence for such problems. An approximation to the fourth derivative $\nabla^4 f(x_c)$ as well, or at least the quantity $\nabla^4 f(x_c)vvvv$, is necessary to obtain better than linear convergence.

This need for fourth order information in order to obtain fast convergence on singular problems is one reason why we will use a fourth order model, rather than a third order model, in our tensor methods for optimization. Other reasons are that a third order model is unbounded below, even though it may have a local minimizer, and that the information that is readily available in an optimization algorithm, namely values of $f(x)$ and $\nabla f(x)$ at previous iterates, naturally supports the use of a fourth order tensor model. Note that these conditions are quite different from the situation for systems of nonlinear equations, where an approximation to $F''(x_*)$ (analogous to $\nabla^3 f(x_*)$) *is* sufficient to produce faster than linear convergence on problems where the Jacobian at the solution is singular, and where only one piece of interpolation information ($F(x)$, analogous to $\nabla f(x)$) is readily available from each previous iterate.

For these reasons, we have based the tensor methods for unconstrained minimization discussed in this paper upon the fourth order tensor model

$$m_T(x_c+d) = f(x_c) + \nabla f(x_c)^T d + \frac{1}{2} d^T \nabla^2 f(x_c) d \quad + \frac{1}{6} T_c \, ddd + \frac{1}{24} V_c \, dddd \qquad (1.5)$$

where by $\nabla f(x_c)$ and $\nabla^2 f(x_c)$ we mean either these analytic derivatives, or finite difference approximations to them, and where $T_c \in R^{n \times n \times n}$ and $V_c \in R^{n \times n \times n \times n}$ are symmetric. (The symmetry of $T_c$ and $V_c$ is another significant difference between tensor models for unconstrained optimization and for nonlinear equations, where $T_c$ is *not* symmetric.) The three-dimensional object $T_c$ and the four-dimensional object $V_c$ are referred to as tensors, hence we call (1.5) a tensor model, and methods based on (1.5) tensor methods. Before proceeding, we define the notation concerning these tensors that is used above and in the remainder of this paper.

**Definition 1.1** Let $T \in R^{n \times n \times n}$. Then for $u, v, w, \in R^n$, $Tuvw \in R$, $Tvw \in R^n$, with

$$Tuvw = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} T[i,j,k] \, u[i] \, v[j] \, w[k],$$

$$Tvw[i] = \sum_{j=1}^{n} \sum_{k=1}^{n} T[i,j,k] \, v[j] \, w[k] \quad , i=1, \cdots, n \; .$$

**Definition 1.2** Let $V \in R^{n \times n \times n \times n}$. Then for $r, u, v, w \in R^n$, $Vruvw \in R$, $Vuvw \in R^n$ with

$$Vruvw = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V[i,j,k,l] \, r[i] \, u[j] \, v[k] \, w[l],$$

$$Vuvw[i] = \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V[i,j,k,l] \, u[j] \, v[k] \, w[l] \quad , i=1, \cdots, n \; .$$

The obvious choices of $T_c$ and $V_c$ in (1.5) are $\nabla^3 f(x_c)$ and $\nabla^4 f(x_c)$; these would make (1.5) the first five terms of the Taylor series expansion of $f$ around $x_c$. We will not consider using the actual higher order derivatives in the tensor model, however, because the cost of doing so would be prohibitive.

In particular, $O(n^4)$ partial derivatives would have to be evaluated or approximated at each iteration; storing these derivatives would take $O(n^4)$ locations; and finding a minimizer or the model would require the solution of a difficult minimization problem in $n$ variables at each iteration. Each of these reasons alone is sufficient to reject this alternative for a general purpose method, although we note that for some functions $f(x)$ with special forms, using analytic higher order information can be viable (see Jackson and McCormick [1986]).

Instead, our new method will choose $T_c$ and $V_c$ in (1.5) to be simple, low-rank symmetric approximations to $\nabla^3 f(x_c)$ and $\nabla^4 f(x_c)$ that are formed from previously calculated function and gradient values. The remainder of this paper will show how we efficiently form and solve such a tensor model, how we incorporate it into a complete unconstrained optimization algorithm, and what the computational performance of this algorithm is. Section 2 describes how we form the tensor model, and shows that this requires only a small multiple of $n^2$ additional arithmetic operations per iteration, and a small multiple of $n$ additional storage locations. In Section 3 we show how we solve this model using only $0(n^2)$ more operations per iteration than the $O(n^3)$ operations that are needed by the standard quadratic model. A full tensor algorithm for unconstrained optimization is presented in section 4. In section 5, we present test results of our tensor method on problems from Moré, Garbow and Hillstorm [1981], and on modifications of these problems constructed so that $\nabla^2 f(x_*)$ is singular. We compare these results to those obtained from a state-of-the-art algorithm that uses the standard quadratic model (1.3) but is identical to the tensor algorithm in virtually all other respects. We briefly summarize our research and comment on possible extensions of this work in Section 6.

We will denote members of a sequence of $n$-vectors $x$ by $\{x_k\}$, where each $x_k \in R^n$, and to avoid ambiguity with this notation, we will continue to denote components of a vector $v \in R^n$ by $v[i] \in R$. We will also abbreviate terms of the form $dd$, $ddd$, and $dddd$ in our tensor models by $d^2$, $d^3$, and $d^4$ respectively.

## 2. Forming the Tensor Model

Now we discuss how we select the tensor terms $T_c \in R^{n\times n\times n}$ and $V_c \in R^{n\times n\times n\times n}$ in the model

$$m_T(x_c+d) = f(x_c)+\nabla f(x_c)^T d+\frac{1}{2}d^T\nabla^2 f(x_c)d+\frac{1}{6}T_c d^3+\frac{1}{24}V_c d^4 . \tag{2.1}$$

We have already stated that $T_c$ and $V_c$ will not contain actual third and fourth derivative information. Instead, we will use the third and the fourth order terms in (2.1) to cause the model to interpolate function and gradient information that has been computed at some previous iterations of the optimization algorithm. In particular, we will select $p$ not necessarily consecutive past iterates $x_{-1}, \cdots, x_{-p}$, and ask that the model (2.1) interpolate $f(x)$ and $\nabla f(x)$ at these points, i.e.

$$f(x_{-k}) = f(x_c)+\nabla f(x_c)^T s_k+\frac{1}{2}s_k^T\nabla^2 f(x_c)s_k+\frac{1}{6}T_c s_k^3+\frac{1}{24}V_c s_k^4, \quad k=1,\cdots,p \tag{2.2a}$$

$$\nabla f(x_{-k}) = \nabla f(x_c)+\nabla^2 f(x_c)s_k+\frac{1}{2}T_c s_k^2+\frac{1}{6}V_c s_k^3, \quad k=1,\cdots,p \tag{2.2b}$$

where

$$s_k = x_{-k}-x_c, \quad k=1,\cdots,p. \tag{2.2c}$$

First we briefly summarize how the past points $x_{-1}, \cdots, x_{-p}$ are selected. Then we discuss how we select $T_c$ and $V_c$ so that (2.2) is satisfied.

The set of past points used in the interpolation conditions (2.2) is selected by the procedure given in Schnabel and Frank [1984]. We always select the most recent previous iterate, and then select each preceding past iterate if the step from it to $x_c$ makes an angle of at least $\theta$ degrees with the subspace spanned by the steps to the already selected, more recent iterates. Values of $\theta$ between 20 and 45 degrees have proven to be best in practice; therefore the selected directions $\{s_k\}$ are strongly linearly independent. This procedure is easily implemented using a modified Gram-Schmidt algorithm. We also set an upper bound

$$p \le n^{1/3} . \tag{2.3}$$

on the number of past points. This bound was motivated by computational experience that showed that using more than about $n^{1/3}$ interpolation conditions rarely helped much, and also by the desire to keep the storage and arithmetic costs of our tensor method low. In fact, however, our computational results will show that the strong linear independence criterion discussed above usually limits $p$ far more severely than (2.3).

Now we discuss how we choose $T_c$ and $V_c$ to satisfy (2.2). First we show that the interpolation conditions (2.2) uniquely determine $Ts_k^3$ and $Vs_k^4$ for each $k = 1, \cdots, p$. Multiplying (2.2b) by $s_k$ gives

$$\nabla f(x_{-k})^T s_k = \nabla f(x_c)^T s_k + s_k^T \nabla^2 f(x_c) s_k^2 + \frac{1}{2} T_c s_k^3 + \frac{1}{6} V_c s_k^4, \quad k=1, \cdots, p \ . \tag{2.4}$$

Let the unknown quantities $\alpha, \beta \in R^p$ be defined by

$$\alpha[k] = Ts_k^3 \ , \tag{2.5a}$$

$$\beta[k] = Vs_k^4 \ , \tag{2.5b}$$

for $k=1, \cdots, p$. Then from (2.2a) and (2.4) we have the following systems of two linear equations in two unknowns for each of the $p$ pairs $\alpha[k]$ and $\beta[k]$ :

$$\frac{1}{2}\alpha[k] + \frac{1}{6}\beta[k] = q_1[k] \ , \tag{2.6a}$$

$$\frac{1}{6}\alpha[k] + \frac{1}{24}\beta[k] = q_2[k] \ , \tag{2.6b}$$

where $q_1, q_2 \in R^p$ are defined by

$$q_1[k] = \nabla f(x_{-k})^T s_k - \nabla f(x_c)^T s_k - s_k^T \nabla^2 f(x_c) s_k,$$

$$q_2[k] = f(x_{-k}) - f(x_c) - \nabla f(x_c)^T s_k - \frac{1}{2} s_k^T \nabla^2 f(x_c) s_k,$$

for $k=1, \cdots, p$, The system (2.6) is nonsingular, so each $\alpha[k]$ and $\beta[k]$ is uniquely determined.

Thus for each $k$, our interpolation conditions uniquely determine $Vs_k^4$ and $Ts_k^3$, and, from (2.2b), leave us with $np$ linear equations in $O(n^4)$ unknowns to determine the $p$ terms $Ts_k^2 + \frac{1}{3} Vs_k^3$. These are the

only remaining interpolation conditions, meaning that the choices of $T$ and $V$ are vastly underdetermined.

We have chosen to select $T$ and $V$ from among the infinite number of possibilities by first choosing the smallest symmetric $V$, in the Frobenius norm, for which

$$Vs_k^4 = \beta[k], \quad k=1, \cdots p$$

where $\beta[k]$ is calculated by (2.6). The rationale behind this choice is to use the smallest fourth order term consistent with the interpolation conditions, thereby modeling as much of the function and gradient information as possible with a third order model. This choice also will give us the necessary fourth order information in the singular case. We then substitute this value of $V$ into (2.2b), obtaining

$$T_c s_k^2 = a_k, \quad k=1, \cdots, p \tag{2.7a}$$

where

$$a_k = 2(\nabla f(x_{-k}) - \nabla f(x_c) - \nabla^2 f(x_c) - \frac{1}{6} V s_k^3), \quad k=1, \cdots, p. \tag{2.7b}$$

This is a set of $np < n^{4/3}$ linear equations in $n^3$ unknowns $T_c[i,j,k]$, $1 \le i,j,k \le n$. Finally we choose the smallest symmetric $T_c$, in the Frobenius norm, which satisfies the equations (2.7). The use of the minimum norm solution here is consistent with the tensor method for nonlinear equations, and will again be a key to the efficiency of our method because it will cause $T_c$ and $V_c$ to have low rank.

The solutions to the minimum norm problems that determine $V_c$ and $T_c$ are given by Theorems 2.2 and 2.3. We note that deriving the minimum norm $T_c$ is much more difficult than in the tensor method for nonlinear equations, because of the symmetry requirement. First we define three and four dimensional rank one tensors, which will feature prominently in these theorems and the remainder of this paper. Also recall that the Frobenius norm of any matrix or tensor $A$, denoted $||A||_F$, is the square root of the sum of the squares of all the elements of $A$.

**Definition 2.1** Let $u,v,w,x \in R^n$. The tensor $T \in R^{n \times n \times n}$ for which $T[i,j,k] = u[i]*v[j]*w[k]$, $1 \le i,j,k \le n$ is called a third order rank one tensor and will be denoted $T = uvw$. The tensor $V \in R^{n \times n \times n \times n}$

for which $V[i,j,k,l]=u[i]*v[j]*w[k]*x[l], 1\le i,j,k,l\le n$ is called a fourth order rank one tensor and will be denoted $V=uvwx$.

**Theorem 2.2** Let $p\le n$, let $s_k\in R^n, k=1, \ldots, p$ with $\{s_k\}$ linearly independent, and let $\beta\in R^p$. Define $M\in R^{p\times p}$ by $M[i,j]=(s_i^T s_j)^4$, $1\le i,j\le p$, and define $\gamma\in R^p$ by $\gamma=M^{-1}\beta$. Then the solution to

$$\underset{V_c\in R^{n\times n\times n\times n}}{\text{minimize}} \; ||V_c||_F \text{ subject to } V_c s_k^4=\beta[k],\ k=1,\cdots,p \text{ and } V_c \text{ symmetric} \tag{2.8}$$

is

$$V_c=\sum_{k=1}^{p}\gamma_k(s_k s_k s_k s_k) \ . \tag{2.9}$$

**Proof.** Let $\hat{V}\in R^{n^4}$ be defined by $\hat{V}^T = (V_c[1,1,1,1], V_c[1,1,1,2], \cdots, V_c[1,1,1,n], V_c[1,1,2,1], \cdots, V_c[1,1,2,n], \cdots, V_c[n,n,n,n])$. Also let $\hat{s}\in R^{p\times n^4}$, be defined so that row k of $\hat{s}$ is equal to $((s_k[1])^4, (s_k[1])^3(s_k[2]), (s_k[1])^3(s_k[3]), \cdots, (s_k[1])^3(s_k[n]), \cdots, (s_k[n])^4)$, i.e. the same order of subscripts as in $\hat{V}$. Then (2.8) is equivalent to

$$\underset{V}{\text{minimize}} \; ||\hat{V}||_2 \text{ subject to } \hat{s}\,\hat{V}=\beta \text{ and } V_c \text{ symmetric} \ ,$$

where $V_c$ is the original form of $\hat{V}$. Since $\{s_k\}$ are linearly independent, $\hat{s}$ has full row rank. The solution to

$$\underset{V}{\text{minimize}} \; ||\hat{V}||_2 \text{ subject to } \hat{s}\,\hat{V}=\beta$$

is $\hat{V}=\hat{s}^T(\hat{s}\hat{s}^T)^{-1}\beta$. By straightforward algebra, $\hat{s}\hat{s}^T=M$. Thus $\hat{V}=\hat{s}^T\gamma$, which by reversing the transformation from $\hat{V}$ back to $V_c$ is equivalent to (2.9). Since $V_c$ is symmetric, it is the solution to (2.8). □

**Theorem 2.3** Let $p\le n$, let $s_k\in R^n$, $k=1,\cdots,p$ with $\{s_k\}$ linearly independent, and let $a_k\in R^n$, $k=1,\cdots,p$. The solution to

$$\underset{T_c \in R^{n\times n\times n}}{\text{minimize}} \; ||T_c||_F \text{ subject to } T_c s_i s_i = a_i,\ i=1,\cdots,p \text{ and } T_c \text{ symmetric} \tag{2.10}$$

is

$$T_c = \sum_{k=1}^{p} b_k s_k s_k + s_k b_k s_k + s_k s_k b_k \tag{2.11}$$

where $b_k \in R^n$, $k=1,\cdots,p$, and $\{b_k\}$ is the unique set of vectors for which (2.11) satisfies $T_c s_i s_i = a_i$, $i=1,\cdots,p$.

**Proof.** First we show that the constraint set in (2.10) is feasible. Let $t_i \in R^n$, $i=1,\cdots,p$, obey

$$t_i^T s_j = \begin{cases} 1, & i=j \\ 0, & i \neq j \end{cases}$$

for $j=1,\cdots,p$. Such vectors $t_i$ are obtainable via a QR factorization of the matrix whose columns are the $s_i$. Then

$$T = \sum_{i=1}^{p} t_i t_i a_i + t_i a_i t_i + a_i t_i t_i - 2(a_i^T s_i)(t_i t_i t_i)$$

is a feasible solution to (2.10).

Dennis and Schnabel [1979] show that if the constraints in (2.10) are satisfiable, then the set of tensors $T_i \in R^{n\times n\times n}$ generated by the procedure $T_0=0$, and for all $j=0,1,2,\cdots$, $T_{2j+1}$ is the solution of

$$\text{minimize } ||T_{2j+1} - T_{2j}||_F \text{ subject to } T_{2j+1} s_i s_i = a_i,\ i=1,\cdots,p \tag{2.12}$$

and $T_{2j+2}$ is the solution of

$$\text{minimize } ||T_{2j+2} - T_{2j+1}||_F \text{ subject to } T_{2j+2} \text{ is symmetric,}$$

has a limit which is the unique solution to (2.10).

Next we show that this limit has the form (2.11) for some set of vectors $\{b_k\}$, by showing that each $T_{2j}$ has this form. This is trivially true for $T_0$. Assume it is true for some fixed $j$, i.e.

$$T_{2j}=\sum_{k=1}^{p} u_k s_k s_k+s_k u_k s_k+s_k s_k u_k \tag{2.13}$$

for some set of vectors $\{u_k\}$. Then from Schnabel and Frank [1984], the solution to (2.12) is

$$T_{2j+1}=T_{2j}+\sum_{k=1}^{p} v_k s_k s_k$$

for some set of vectors $\{v_k\}$. Thus

$$T_{2j+2}=T_{2j}+\frac{1}{3}(\sum_{k=1}^{p} v_k s_k s_k+s_k v_k s_k+s_k s_k v_k)$$

$$=\sum_{k=1}^{p}(u_k+\frac{v_k}{3})s_k s_k+s_k(u_k+\frac{v_k}{3})s_k+s_k s_k(u_k+\frac{v_k}{3})\ .$$

which again has the form (2.13). Thus by induction the solution $T_c$ to (2.10) must have the form (2.11) for some set of vectors $\{b_k\}$.

Finally we show that the set of vectors $\{b_k\}$ for which $T_c$ given by (2.11) satisfies

$$T_c s_i s_i=a_i,\quad i=1,\cdots,p \tag{2.14}$$

is unique. This will mean that the equations (2.11) and (2.14) uniquely determine the solution to (2.10). Substituting (2.11) into (2.14) gives a system of $np$ linear equations in $np$ unknowns, where the matrix is a function of the $\{s_k\}$, the unknowns are the elements of the $\{b_k\}$, and the right hand side consists of the elements of the $\{a_k\}$. Since we showed above that (2.10) is feasible for any $\{a_k\}$, the above derivation and the theory of Dennis and Schnabel [1979] imply that for any set $\{s_k\}$, this linear system has at least one solution for any right hand side. Thus the linear system must be nonsingular and have a unique solution. This means that the set of vectors $\{b_k\}$ is uniquely determined, and completes the proof. □

Theorems 2.2 and 2.3 show that $T_c$ and $V_c$ determined by the minimum norm problems (2.10) and (2.8) have rank $2p$ and $p$, respectively. This is the key to making the tensor model efficient to store and solve. However, while the proof of Theorem 2.3 shows constructively that there is a unique $T_c$ of the form (2.11) that satisfies (2.10), it does not give an efficient algorithm for finding it, since the proof involves solving a system of $np$ linear equations in $np$ unknowns. We now present an efficient method

for finding $T_c$.

Substituting (2.11) into (2.14) gives the equations

$$a_i=(\sum_{k=1}^{p} b_k s_k s_k+s_k b_k s_k+s_k s_k b_k)s_i s_i$$

$$=\sum_{k=1}^{p} b_k (s_k^T s_i)^2+2\sum_{k=1}^{p} s_k (s_k^T s_i)(b_k^T s_i) \ ,$$

$i=1,\cdots,p$ in the unknowns $\{b_k\}$. We can write these equations in the matrix form

$$A = B\,N + 2\,S\,M \tag{2.15}$$

where $A\in R^{n\times p}$, with column $k$ of $A=a_k$, $B\in R^{n\times p}$, with column $k$ of $B=b_k$, $S\in R^{n\times p}$, with column $k$ of $S=s_k$, and $N$, $M\in R^{p\times p}$ with $N_{ij}=(s_i^T s_j)^2$ and $M_{ij}=(s_i^T s_j)(b_i^T s_j)$, $1\le i,j\le p$. Note that $B$ contains the unknowns, that $M$ is a linear function of these unknowns, and that $A$, $N$, and $S$ are known. Pre-multiplying both sides of (2.15) by $S^T$ gives

$$[S^T A] = [S^T B]N + 2[S^T S]M. \tag{2.16}$$

Defining $x_{ij}=b_j^T s_i$, $1\le i,j\le p$, we can rewrite (2.16) in the form of $p^2$ linear equations in the $p^2$ unknowns $x_{ij}$

$$\begin{bmatrix} s_1^T a_1 \\ s_1^T a_2 \\ \vdots \\ s_p^T a_p \end{bmatrix} = \begin{bmatrix} N & & & \\ & N & & \\ & & \ddots & \\ & & & N \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{pp} \end{bmatrix} + \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1p} \\ w_{p1} & & & w_{pp} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{pp} \end{bmatrix} \tag{2.17}$$

where each $w_{ij}$ in the second matrix of (2.17) is a $p$-vector given by $w_{ij}=[(s_i^T s_1)(s_1^T s_j), (s_i^T s_2)(s_2^T s_j), \cdots, (s_i^T s_p)(s_p^T s_j)]^T$. The only unknowns in (2.17) are the $x_{ij}$, so we can solve (2.17) for $x_{ij}$, and then compute $M$ by

$$M_{ij} = (s_i^T s_j)(b_i^T s_j) = (s_i^T s_j)x_{ij}.$$

Finally, from (2.15), we can compute $B$ by

$$B = (A - 2SM)N^{-1} \ . \tag{2.18}$$

Note that $N$ is symmetric and positive definite since the $\{s_k\}$ are linearly independent.

We conclude this section by summarizing the costs to form and store the tensor model. The dominant costs of the process for calculating $T_c$ that is summarized in equations (2.17) and (2.18) are $np^2$ each multiplications and additions for calculating $S^TA$, the same cost for calculating $S*M$, the same cost again for the backsolves in (2.18), roughly $np^2/2$ each multiplications and additions for calculating $s_i^Ts_j$ for $1 \le i \le j \le p$, and $p^6/3$ each multiplications and additions for solving the system (2.17), for a total of $(7/2)np^2 + p^6/3$ each multiplications and additions. Since $p \le n^{1/3}$, these are all $O(n^2)$ or less. The additional costs of forming $V_c$ are negligible, at most $O(p^3)$. In addition, the cost of forming the interpolation equations (2.2) includes the multiplication $\nabla^2 f(x_c)s_k$ for $k=1, \cdots, p$ which requires $n^2p$ each multiplications and additions. This is generally the largest cost of forming the tensor model. The Gram-Schmidt process for selecting the $\{s_k\}$ requires about $n^{5/3}$ arithmetic operations if $n^{1/3}$ vectors are considered. In summary, only a small multiple of $n^2$ additional arithmetic operations are required to form the model. We will see in section 5 that usually $p=1$, so that the total additional cost per iteration is usually $n^2 + n^{5/3} + O(n)$ each additions and multiplications per iteration.

The storage required for forming and storing the tensor model is also small. The tensor terms $T_c$ and $V_c$ themselves require only the storage of the vectors $b_k$ and $s_k$, which takes $2np \le 2n^{4/3}$ storage locations. In addition, the model formation process requires at most $2n^{4/3}$ storage locations for storing $n^{1/3}$ each past iterates and their gradients, $np \le n^{4/3}$ storage locations for intermediate quantities in (2.18), and $p^4 \le n^{4/3}$ storage locations for the factorization in solving (2.17). Thus the total additional storage is at most $6n^{4/3}$.

## 3. Solving the Tensor Model

In Section 2 we showed how to find a rank $2p$ tensor $T_c$ of the form (2.11), and a rank $p$ tensor $V_c$ of the form (2.9), for which the tensor model (2.1) interpolates the function and gradient values at $p$ ($\leq n^{1/3}$) past iterates. Substituting these values of $T_c$ and $V_c$ into (2.1), the tensor model has the form

$$m_T(x_c+d) = f(x_c) + \nabla f(x_c)^T d + \frac{1}{2} d^T \nabla^2 f(x_c) d + \frac{1}{2} \sum_{k=1}^{p} (b_k^T d)(s_k^T d)^2 + \frac{1}{24} \sum_{k=1}^{p} \gamma_k (s_k^T d)^4 . \quad (3.1)$$

In this section we show how to efficiently find a minimizer of this model. Although equation (3.1) is a fourth order polynomial in $n$ variables, we will show how to reduce its minimization to the minimization of a fourth order polynomial in $p$ variables plus a quadratic in $n-p$ variables. For conciseness we use the notation $g=\nabla f(x_c)$ and $H=\nabla^2 f(x_c)$ for the remainder of this section.

Let $S \in R^{n \times p}$, where column $k$ of $S$ is $s_k$, and the $\{s_k\}$ are linearly independent. Also let $Z \in R^{n \times (n-p)}$ and $W \in R^{n \times p}$ have full column rank and satisfy $Z^T S=0$ and $W^T S=I$ respectively. ($Z$ and $W$ can be calculated through the QR factorization of $S$; the efficient implementation of the operations involving $Z$ and $W$ is discussed later in this section.) Then we can write $d$ in (3.1) in the form

$$d = Wu + Zt \quad (3.2)$$

where $u \in R^p$, $t \in R^{n-p}$. Substituting (3.2) into (3.1) gives

$$m_T(x_c+Wu+Zt) = f(x_c) + g^T Wu + g^T Zt + \frac{1}{2} u^T W^T H W u \quad (3.3)$$

$$+ u^T W^T H Z t + \frac{1}{2} t^T Z^T H Z t + \frac{1}{2} \sum_{k=1}^{p} u_k^2 (b_k^T W u + b_k^T Z t) + \frac{1}{24} \sum_{k=1}^{p} \gamma_k u_k^4 .$$

Equation (3.3) is a quadratic with respect to $t$. Therefore, for the tensor model to have a minimizer, $Z^T H Z$ must be positive definite and the derivative of the model with respect to $t$ must be 0, i.e.

$$Z^T g + Z^T H Z t + Z^T W^T H u + \frac{1}{2} Z^T \sum_{i=1}^{p} b_i u_i^2 = 0 \quad (3.4)$$

which yields

$$t=-(Z^T HZ)^{-1}Z^T(g+HWu+\frac{1}{2}\sum_{i=1}^{p}b_i u_i^2). \tag{3.5}$$

Thus, if $Z^T HZ$ is positive definite, substituting (3.5) into (3.3) reduces the problem of minimizing the tensor model to finding a minimizer of

$$\hat{m}_T(u)=f+g^T Wu+\frac{1}{2}u^T W^T HWu+\frac{1}{2}\sum_{i=1}^{p}u_i^2(b_i^T Wu)+\frac{1}{24}\sum_{i=1}^{p}\gamma_i u_i^4 \tag{3.6}$$

$$-\frac{1}{2}(g+HWu+\frac{1}{2}\sum_{i=1}^{p}b_i u_i^2)^T Z(Z^T HZ)^{-1}Z^T(g+HWu+\frac{1}{2}\sum_{i=1}^{p}b_i u_i^2)$$

which is a fourth degree polynomial in $p$ variables. If (3.6) has a minimizer $u_*$, then the minimizer of the original tensor model (3.1) is given by $d_* = Wu_* + Zt_*$, where $t_*$ is determined by setting $u = u_*$ in (3.5). Note that this process is well defined even if $H$ is singular, as long as $Z^T HZ$ is nonsingular and positive definite. This is possible if $\text{rank}(H) \geq n-p$.

There are several possible difficulties with this process. First, (3.6) may have multiple minimizers. If $p=1$, we can find the minimizers analytically, and if there are two, we choose the value of $u_*$ that is in the same valley of the function $\hat{m}_T(u)$ as $u=0$. This choice can be shown to guarantee that there is a (nonlinear) descent path from $x_c$ to $x_c+d_*$ for the model $m_T(x_c+d)$. If $p>1$ we minimize (3.6) with a standard unconstrained minimization code (starting from $u=0$) and use the minimizer it returns. We have found that these procedures generally produce a desirable minimizer.

Secondly, the tensor model may not have a minimizer, either because $Z^T HZ$ is not positive definite, or because (3.6) has no minimizer when $Z^T HZ$ is positive definite. Finally, even if (3.6) has a minimizer $d_*$, $x_c+d_*$ may not be an acceptable next iterate. These difficulties are addressed by using a global strategy.

We have tried both line search and trust region global strategies in conjunction with our tensor method. The line search strategy we used is simple : if (3.6) has a minimizer $d_*$ which is in a descent direction, but $x_c+d_*$ is not an acceptable next iterate, we set $x_+ = x_c+\lambda d_*$ for some $\lambda \in (0,1]$ using a standard line search. If (3.6) has no minimizer, or $d_*$ is not in a descent direction, we find the next iterate by

using a line search algorithm based on the standard quadratic model (1.3). The tensor method based on this strategy has performed quite well (see Section 5), but we find that about 40% of the iterations cannot use the tensor model. In order to make fuller use of the tensor model, we have also tried a trust region strategy, which is the method that we concentrate on in this paper.

The trust region method approximately solves the problem

$$\underset{d \in R^n}{\text{minimize}} \; m_T(x_c + d) \quad \text{subject to } \|d\|_2 \leq \delta \tag{3.7}$$

where $\delta \in R$ is the trust radius that is adjusted at each iteration. This is a standard type of approach for unconstrained optimization, see for example Fletcher [1980], Dennis and Schnabel [1983]. Efficient methods exist for solving the trust region problem with quadratic models (see e.g. Moré and Sorensen [1983], but it is quite difficult to extend them to the tensor model. For this reason, in order to test the trust region tensor method approach initially, we used a penalty method to solve (3.7). This means that we solve (3.7) by solving a sequence of unconstrained optimization problems of the form

$$\underset{d \in R^n}{\text{minimize}} \; m_T(x_c + d) + \sigma (d_p^T d_p - \delta^2)^2 \tag{3.8}$$

for increasing positive values of the scalar $\sigma$. (The details of selecting $\sigma$ are given in Chow [1989].) As in most trust region algorithms, we only solve (3.7) approximately; in our implementation we stop when a solution $d_*(\sigma)$ to (3.7) satisfies $\|d_*(\sigma)\| \in [0.95\delta, 1.05\delta]$. This means that $\sigma$ does not grow unboundedly, and in practice a small number of problems of the form (3.8) are solved per iteration. The penalty approach is only intended for initial test purposes, because it increases the cost of each iteration considerably due to the cost of solving (3.8), although it does not increase the cost in function and derivative evaluations. We will see that our best results so far have been obtained when $p$ is constrained to be 1 at each iteration; an efficient but complicated method for solving (3.7) in this case is given in Chow [1989].

Finally, we discuss the costs of solving the tensor model. The main additional calculations in finding a minimizer of the tensor model, in comparison to minimizing a standard quadratic model, are the calculations involving the matrices $Z$ and $W$. These are performed by calculating the decomposition $S$ =

$Q \cdot R$, where $Q \in R^{n \times n}$ is an orthogonal matrix that is the product of $p$ Householder transformations, and $R \in R^{n \times p}$ consists of an upper triangular matrix $R_1$ in its first $p$ rows, and is 0 otherwise. ($Q$ is not actually formed, rather the $p$ $n$-vectors that determine the $p$ Householder transformations are stored, see e.g. Stewart [1970].) Also let $\hat{R} \in R^{n \times p}$ consist of $(R_1)^{-1}$ in its first $p$ rows and 0 otherwise. Then $W = Q \cdot \hat{R}$, so for any $v \in R^n$ we can calculate $W^T v$ in $2np$ each multiplications and additions by applying the $p$ Householder transformations for $Q^T$ followed by $O(p^2)$ operations to apply $(R_1)^1$. Similarly $Z = Q \cdot \hat{I}$ where $\hat{I}$ is 0 in its first $p$ rows and the identity matrix in its bottom $n-p$ rows. Thus for any $v \in R^n$ we can calculate $Z^T v$ in $2np$ each multiplications and additions by applying $Q^T$ and then $\hat{I}$. Using these techniques, it is straightforward to verify that all the calculations in the tensor method that involve $Z$ and $W$, as well as the $QR$ decomposition of $S$, can be performed in $4n^2p + O(np^2)$ each multiplications and additions per iteration; the leading term comes from calculating $HQ$ and then $Q^T HQ$.

The other costs of minimizing the tensor model are $(n-p)^3/6$ each multiplications and additions for the factorization of $Z^T HZ$, and the cost of minimizing the fourth order polynomial in $p$ variables (3.6), which is negligible in comparison to the $O(n^3)$ cost, especially when $p = 1$. Thus the total cost of minimizing the tensor model is only a small multiple of $n^2p$ operations more than the $n^3/6$ cost of finding a minimizer of a standard quadratic model. Since $p \leq n^{1/3}$ and we will see that usually $p = 1$, this is a very small additional cost.

At many iterations, the tensor model has a minimizer which is accepted as the next iterate, so these are the only costs of solving the tensor model. If a global strategy is needed, then the line search described above can be implemented with about the same cost as for a standard quadratic model, since given the factorization of $Z^T HZ$ we can also factor $H$ using only $O(n^2p)$ additional operations. In the case $p=1$, the trust region strategy can also be implemented as efficiently as in the quadratic case, i.e. requiring the minimization of the tensor model at each inner iteration, by using the techniques in Chow [1989]. The penalty approach is more expensive but is only intended for test purposes.

## 4. The Complete Tensor Method Algorithm

An outline of the complete tensor method algorithm that we used in our computational tests is given in Algorithm 4.1. The remainder of this section comments on several aspects of this algorithm that have not yet been discussed.

**Algorithm 4.1 -- An iteration of the tensor method.** Given $x_c$, $f(x_c)$, $\delta_c$ :

1. Calculate $\nabla f(x_c)$ and decide whether to stop. If not:
2. Calculate $\nabla^2 f(x_c)$.
3. Select $p$ past points to use in the tensor model from among the $n^{1/3}$ most recent past points.
4. Calculate the terms $T_c$ and $V_c$ in the tensor model, so that the tensor model interpolates $f(x)$ and $\nabla f(x)$ at all the points selected in step 3.
5. Find a potential acceptable next iterate $x_c+d_T$ and a potential new trust radius $\delta_T$ using the tensor model and a trust region strategy.
6. Find a potential acceptable next iterate $x_c+d_N$ and a potential new trust radius $\delta_N$ using the quadratic model and a trust region strategy.
7. If $f(x_c+d_T) <= f(x_c+d_N)$

   then set $x_+ = x_c+d_T$ and $\delta_+ = \delta_T$

   else set $x_+ = x_c+d_N$ and $\delta_+ = \delta_N$.
8. Set $x_c=x_+$, $f(x_c)=f(x_+)$, $\delta_c = \delta_+$, go to step 1.

The most important feature of Algorithm 4.1 that has not been previously discussed is that at each iteration, we calculate a potential new iterate based on the quadratic model, as well as a potential new iterate based on the tensor model. This means we perform a full global strategy using each model, resulting in two points $x_c+d_T$ and $x_c+d_N$ which both give sufficient decrease in $f(x)$ to be acceptable as the next iterate. Then we choose the one with the lower function value as the next iterate. Even though this strategy increases the cost of each iteration by at least one function evaluation (since it is necessary to evaluate $f(x)$ at both $x_c+d_T$ and $x_c+d_N$, and maybe at some unsuccessful trial points, in the global

strategies), we have found that this approach substantially improves the efficiency of our tensor method as measured in function and derivative evaluations, as well as in iterations. We have not yet found a way to achieve the same efficiency without requiring the use of both models at each iteration.

Finally we discuss some details of the steps of Algorithm 4.1. In steps 1 and 2, the gradient and Hessian are approximated by finite differences using Algorithms A5.6.3 and A5.6.2 in Dennis and Schnabel [1983], respectively. The algorithm stops if $||\nabla f(x_c)||_2 \leq 10^{-5}$ or $||d_c||_2 \leq 10^{-10}$. Step 3 was discussed in Section 2; 45 degrees is used for the angle $\theta$ mentioned there. The procedures for calculating $T_c$ and $V_c$ in step 4 also were discussed in Section 2.

In step 5, we first determine whether the tensor model has an acceptable minimizer within the trust region, and if so, we select this point as the solution to step 5. Otherwise we solve the trust region problem (3.7) by a penalty method, as discussed in Section 3, resulting in a candidate step $d$. Then we decide whether to accept $x_c+d$ as the solution to step 5, update the trust radius, and possibly repeat this process until an acceptable point $x_c+d_T$ is found. In step 6, we follow the exact same procedure except that we only use the first three terms of the model. The procedure for determining whether the candidate step is acceptable in these trust region algorithms, and for updating the trust region, is identical to Algorithm A6.4.5 in Dennis and Schnabel [1983], except that : 1) every occurrence of *initslope* is changed to $\Delta fpred$, where $\Delta fpred$ is the difference of the values of the model being used (tensor or quadratic) at the candidate point and at $x_c$ ; 2) steps (9c.1-2) of Algorithm A6.4.5 are replaced by setting $\Delta fpred$ to this same value.

## 5. Test Results

We have tested the tensor algorithm described in Section 4 on a variety of nonsingular and singular problems. We compared it to an algorithm that is identical except that the third and fourth order terms $T_c$ and $V_c$ are always zero. That is, the comparison algorithm is a finite difference Newton's method, whose global strategy is a trust region problem solved by a penalty method. In this section we summarize our test results. The details of our computational results are provided in the appendix. All our computations were performed on a MIPS computer, using double precision arithmetic.

First we tested our algorithms on the set of unconstrained optimization problems in Moré, Garbow and Hillstrom [1981]. All these problems except the Powell singular problem have $\nabla^2 f(x_*)$ nonsingular. The dimensions of the problems range from 2 to 30.

Then we created singular test problems by modifying the nonsingular test problems of Moré, Garbow and Hillstrom [1981]. All of the unconstrained optimization test problems in that paper are obtained by taking a system of nonlinear equations

$$F(x)^T = (f_1(x), \cdots, f_m(x)) \tag{5.1}$$

where $m \geq n$ and each $f_i : R^n \rightarrow R$, and setting

$$f(x) = F(x)^T F(x) = \sum_{i=1}^{m} f_i^2(x). \tag{5.2}$$

In most cases, $F(x) = 0$ at the minimizer $x_*$, and $F'(x_*)$ is nonsingular. In these cases, Schnabel and Frank [1984] showed how to create singular systems of nonlinear equations from (5.1), by forming

$$\hat{F}(x) = F(x) - F'(x_*)A(A^TA)^{-1}A^T(x-x_*), \tag{5.3}$$

where $A \in R^{n \times k}$ has full column rank with $1 \leq k \leq n$. Thus $\hat{F}(x_*)=0$ and $\hat{F}'(x_*)$ has rank $n-k$. To create a singular unconstrained optimization problem, we simply define the function

$$\hat{f}(x) = \tfrac{1}{2}\hat{F}(x)^T\hat{F}(x). \tag{5.4}$$

From (5.4) and $\hat{F}(x_*)=0$, we have $\nabla\hat{f}(x_*) = \hat{F}'(x_*)^T\hat{F}(x_*) = 0$. From

$$\hat{F}'(x_*) = F'(x_*)\,[I - A(A^TA)^{-1}A^T] \tag{5.5}$$

and

$$\nabla^2\hat{f}(x_*) = \hat{F}'(x_*)^T\hat{F}'(x_*) + \sum_{i=1}^{m} f_i(x_*)\nabla^2 f_i(x_*) = \hat{F}'(x_*)^T\hat{F}'(x_*) \tag{5.6}$$

we know that $\nabla^2\hat{f}(x_*)$ has rank $n-k$.

By using (5.3-4), we created two sets of singular problems, with $\nabla^2\hat{f}(x_*)$ having rank $n-1$ and $n-2$, by using

$$A \in R^{n\times 1} \quad , \quad A^T = (1,1,\ldots,1)$$

and

$$A \in R^{n\times 2} \quad , \quad A^T = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & -1 & 1 & -1 & \cdots & (-1)^n \end{bmatrix}$$

respectively. We tested our methods on the singular versions of all the nonsingular problems except the Gaussian function and the Gulf research and development function, which we excluded because their nonsingular versions never converged to a minimizer using either standard or tensor methods.

Our computational results for the sets of test problems whose Hessians at the minimizers have ranks $n$, $n-1$, and $n-2$ are summarized in Tables 5.1-5.3, and given in detail in Appendices A1-A3, respectively. For each problem set, Tables 5.1-5.3 compare the performance of the standard method to two versions of the tensor method: the one described in Sections 2-4 where the number of past points interpolated, $p$, is selected at each iteration to be between 1 and $n^{1/3}$, and a second version where $p$ is restricted to be 1 at all iterations. We tested the second version because we observed that the first version generally chose $p = 1$ anyhow, and because the tensor method is considerably simpler to implement, and is cheaper in terms of storage and cost per iteration, when $p = 1$.

Tables 5.1-5.3 summarize the comparative costs of the standard and tensor methods using ratios of two measures: iterations, and function and derivative evaluations. The iteration ratio is the total number of iterations required by the tensor method on all problems that were successfully solved by both

methods, divided by the total number of iterations required by the standard method on these problems. The second ratio is based upon the total number of function evaluations required to solve each problems, including those for finite difference gradients and Hessians (i.e. we count $n$ function evaluations per gradient evaluation and $(n^2+3n)/2$ function evaluations per Hessian evaluation). The ratio reported is the total of these numbers for the tensor method over all problems that were successfully solved by both methods, divided by the total of these numbers for the standard method over the same problems. Tables 5.1-5.3 also contain, for that test set, the number of problems where the performance of the tensor method was better, worse, or the same as the standard method. Here better is defined as at least 5% better in the function evaluation measure, worse is defined as at least 5% worse in the function evaluation measure, and the remaining problems are considered the same.

The statistics in Tables 5.1-5.3 only pertain to test problems that were solved successfully by *both* the standard method and the tensor method. Table 5.4 shows, for each test set, how many problems were solved successfully by the tensor method but not by the standard method, and vice versa.

In summary, Tables 5.1-5.4 show that both the $p \geq 1$ and the $p=1$ versions of the tensor method have a large advantage, in both reliability and efficiency, over the standard method on all three test sets. In each of the six comparisons, a substantial portion of the test problems (between 16% and 22%) are solved by the tensor method and not the standard method, while only two problems in the nonsingular sets and none in the singular sets are solved by the standard method and not the tensor method. In addition, on the problems solved by both methods (between 43 and 50 problems in each of the six cases), the average cost of the tensor method, measured in iterations or function and derivative evaluations, is generally slightly less than half of the cost of the standard method. Finally, the improvements by the tensor method are quite consistent. Totaling all our tests, the tensor method is worse than the standard method in 8% of the test cases (28 of 352), better in 87.5% (308 of 352), and the same in 4.5% (16 of 352).

The performances of the version of the tensor method which constrains $p$ to be 1 and the version that allows $p$ to be between 1 and $n^{1/3}$ are rather similar overall, with the $p=1$ version actually

**Table 5.1 -- Summary of Test Results for Nonsingular Problems**

| method | tensor/std.(itn) | tensor/std.(fcn) | tensor better | std. better | tie |
|---|---|---|---|---|---|
| $p=1$ | 0.496 | 0.580 | 36 | 5 | 3 |
| $p\geq 1$ | 0.468 | 0.488 | 34 | 5 | 4 |

**Table 5.2 -- Summary of Test Results for Rank n-1 Singular Problems**

| method | tensor/std.(itn) | tensor/std.(fcn) | tensor better | std. better | tie |
|---|---|---|---|---|---|
| $p=1$ | 0.465 | 0.400 | 42 | 3 | 2 |
| $p\geq 1$ | 0.479 | 0.466 | 40 | 5 | 2 |

**Table 5.3 -- Summary of Test Results for Rank n-2 Singular Problems**

| method | tensor/std.(itn) | tensor/std.(fcn) | tensor better | std. better | tie |
|---|---|---|---|---|---|
| $p=1$ | 0.449 | 0.390 | 45 | 1 | 3 |
| $p\geq 1$ | 0.489 | 0.466 | 43 | 5 | 2 |

**Table 5.4 -- Number of Problems Solved by Tensor/Standard Method Only**

| method | nonsingular | singular(rank $n-1$) | singular(rank $n-2$) |
|---|---|---|---|
| $p=1$ | 13/2 | 9/0 | 13/0 |
| $p\geq 1$ | 11/2 | 12/0 | 10/0 |

performing somewhat better overall on the singular test sets and the $p\geq 1$ version performing somewhat better on the nonsingular test set. One reason for their similarity is that even when we allow $p>1$, we have found that our criterion for selecting past iterates to interpolate generally results in $p=1$. Over all our test problems, we found that the $p\geq 1$ method selected $p=1$ 85% of the time, $p=2$ 15%, and $p>2$ 0.35%. Thus it appears that the advantages of the tensor method may be achieved by using $p=1$, which would mean that the extra cost of storing and forming the tensor model would be very low, and that the method would be quite simple to implement. In particular, using $p=1$ has the advantage that the formulas

for $T_c$ and $V_c$ are readily available in closed form in terms of the function and derivative values being interpolated, and that solving the tensor model reduces to minimizing a fourth order polynomial in one variable which also can be done in closed form.

In our tests, the global portion of the tensor method (steps 5-7 of Algorithm 4.1) selected the step from the quadratic model about 20% of the time on the average. While this is a rather small percentage, the performance of the tensor method is improved significantly by allowing this possibility.

We do not claim to fully understand why the tensor method performs so much more efficiently and reliably than a state of the art standard method in our tests. What is especially surprising is that the improvements are attained by incorporating a small amount of extra information, usually just the function in gradient from the previous iterate, into the model. Apparently, having a more accurate model in the direction of the previous step is especially useful in practice.

The computational advantage of the tensor method is probably not due to an improved rate of convergence, except when rank($\nabla^2 f(x_*)$) = $n-1$. In particular, when $\nabla^2 f(x_*)$ is nonsingular and $n>1$, it is highly unlikely that the convergence rate of the tensor method is different than the quadratic rate of the standard method. (It is easy to show that the tensor method is at least quadratically convergent in this case because the influence of the tensor terms vanishes asymptotically.) In the case when $\nabla^2 f(x_*)$ has rank $n-1$, we conjecture that the convergence rate of the tensor method is again better than the linear convergence of the standard method, as was shown by Frank [1984] for the tensor method for nonlinear equations. We have not yet attempted to prove this, except in the case $n=1$ where it is straightforward to show that the tensor method converges with order $\frac{1+\sqrt{7}}{3} \approx 1.2$ (Chow [1989]). We did measure the ratios of the errors of successive iterates on our test problems with rank $\nabla^2 f(x_*) = n-1$. An example is given in Table 5.5. We see that the standard method converges linearly with constant = 2/3, as predicted by the theory, and that the tensor method appears to be converging faster than linearly. (An interesting feature of this example is that iterations 2 and 5 of the tensor method increase the error in $x$, even though the function value decreases. We noticed such behavior by the tensor method on several test problems,

although for most it did not occur.) When rank($\nabla^2 f(x_*)$) = $n-2$, the tensor method does not have enough information to prove a faster than linear convergence rate, since it usually uses $p$ =1.

---

**Table 5.5 -- Speed of Convergence on a Typical Problem with Rank $\nabla^2 f(x_*) = n-1$**

(Singular version of variably dimensioned function, $n$=10, started from $x_0$, using $p$=1)

Numbers in the second and the third columns are $\frac{||x_k - x_*||}{||x_{k-1} - x_*||}$

| Iteration (k) | Tensor Method | Standard Method |
|---|---|---|
| 1 | 0.825 | 0.825 |
| 2 | 61.73 | 0.825 |
| 3 | 0.028 | 0.825 |
| 4 | 0.104 | 0.668 |
| 5 | 7.860 | 0.776 |
| 6 | 0.033 | 0.647 |
| 7 | 0.665 | 0.666 |
| 8 | 0.665 | 0.665 |
| 9 | 0.600 | 0.666 |
| 10 | 0.635 | 0.666 |
| 11 | 0.664 | 0.666 |
| 12 | 0.654 | 0.667 |
| 13 | 0.436 | 0.667 |
| 14 | 0.511 | 0.667 |
| 15 | 0.120 | 0.666 |
| 16 | 0.058 | 0.666 |
| 17 | | 0.666 |
| 18 | | 0.666 |
| 19 | | 0.666 |
| 20 | | 0.665 |
| 21 | | 0.665 |
| 22 | | 0.664 |
| 23 | | 0.664 |
| 24 | | 0.667 |

---

Finally, we have also implemented a line search version of the tensor method and compared it to an algorithm using the standard quadratic model and the same line search. We found that, on the average, the performances of the line search and trust region versions of the quadratic model algorithms were very similar, and that the line search version of the tensor method was almost 15% less efficient than the trust region tensor method. (See Chow [1989] for details.) We observed that the global strategy is only able to use a tensor method step about 60% of the time in the line search tensor method versus about 80% of the time in the trust region version. This may be related to the difference in their performances. But the line search tensor method still improves by a large amount over the standard method.

## 6. Summary and Future Research Directions

We have presented a tensor method for unconstrained optimization that bases each iteration upon a fourth order model of the objective function. This model interpolates the function value, gradient, and Hessian of the objective function at the current iterate, and forms its third and fourth order terms by interpolating values of the function and gradient at previous iterates. The costs of storing, forming, and using the model are not significantly more than for a standard method that uses a quadratic Taylor series model.

The computational results of Section 5 show that the tensor method is substantially more reliable and efficient than the corresponding standard method on both the nonsingular and singular problems that we tested. This experience indicates that the tensor method may be preferable to methods available in software libraries for solving small to medium sized unconstrained optimization problems, in cases when analytic or finite difference Hessian matrices are available. Obviously, more computational experience is necessary to determine this conclusively. To facilitate this process, we are developing a software package that implements a tensor method for unconstrained optimization using analytic or finite difference second derivatives, and will make it available shortly. Our software package restricts $p$, the number of past iterates whose function and gradient values are interpolated at each iteration, to be one. The reasons for this choice are that our computational results show that the tensor method with $p=1$ is generally about as

effective as the method that allows $p \geq 1$, and that the method is considerably simpler and cheaper to implement in this case. Initially it will use a line search rather than a trust region, because the line search tensor method is currently much easier to understand, and much faster on small, inexpensive problems, than the trust region version, while still leading to large savings in iterations and function and derivative evaluations on our test problems. A trust region version may be added to the package later.

Several interesting research topics remain concerning the tensor method described in this paper. As indicated above, the development of a simple, efficient method for approximately solving the trust region problem using the tensor method would be very useful. Chow [1989] has developed a fairly efficient, but complex method for solving the trust region problem (3.7) when $p=1$; the question of how to solve this problem efficiently when $p>1$ remains open. It would also be nice to develop an effective global strategy that does not require the determination of the step using *both* the tensor model and the quadratic model at each iteration. Finally, as we mentioned in Section 5, the local convergence analysis in the case $n>1$ remains open.

The standard and tensor methods discussed in this paper both assume that the analytic or finite difference Hessian is available at each iteration. Often in practical applications, however, the analytic Hessian is not available, and it is expensive to calculate by finite differences, so secant (quasi-Newton) methods are used instead. These methods are based on a quadratic model that is formed solely from function and gradient values at the iterates (see e.g. Dennis and Moré [1977], Fletcher [1980], Dennis and Schnabel [1983]). We are developing a secant tensor method for unconstrained optimization that bases each iteration upon a fourth order model that is also determined solely from function and gradient values at the iterates. This work is described in Chow [1989] and in a forthcoming paper.

## 7. References

T. Chow [1989], "Derivative and secant tensor methods for unconstrained optimization", Ph.D. Thesis, Department of Computer Science, University of Colorado at Boulder.

J. E. Dennis Jr. and J. J. Moré [1977], "Quasi-Newton methods, motivation and theory", *SIAM Review* 19, pp. 46-89.

J. E. Dennis Jr. and R. B. Schnabel [1983], *Numerical Methods for Nonlinear Equations and Unconstrained Optimization,* Prentice-Hall, Englewood Cliffs, New Jersey.

J. E. Dennis Jr. and R. B. Schnabel [1979], "Least change secant updates for quasi-Newton methods", *SIAM Review* 21, pp. 443-459.

R. Fletcher [1980], *Practical Method of Optimization, Vol 1, Unconstrained Optimization,* John Wiley and Sons, New York.

P. D. Frank [1984], "Tensor methods for solving systems of nonlinear equations", Ph.D. Thesis, Department of Computer Science, University of Colorado at Boulder.

P. E. Gill, W. Murray, and M. H. Wright [1981], *Practical Optimization,* Academic Press, London.

A. O. Griewank and M. R. Osborne [1983], "Analysis of Newton's method at irregular singularities", *SIAM Journal on Numerical Analysis* 20, pp. 747-773.

R. H. F. Jackson and G. P. McCormick [1986], "The polyadic structure of factorable function tensors with application to high-order minimization techniques", *Journal of Optimization Theory and Applications* 51, pp. 63-94.

J. J. Moré B. S. Garbow, and K. E. Hillstrom [1981], "Testing unconstrained optimization software", *ACM Transactions on Mathematical Software* 7, pp. 17-41.

J. J. Moré and D. C. Sorensen [1983], "Computing a trust region step", *SIAM Journal on Scientific and Statistical Computing* 4, pp. 553-572.

R. B. Schnabel and P. Frank [1984], "Tensor methods for nonlinear equations", *SIAM Journal on Numerical Analysis* 21, pp. 815-843.

R. B. Schnabel and P. Frank [1987], "Solving systems of nonlinear equations by tensor methods," *The State of the Art in Numerical Analysis,* A. Iserles and M.J.D. Powell, eds., Clarendon Press, Oxford, pp. 245-271.

G. W. Stewart [1970], *Introduction to Matrix Computations,* Academic Press, New York.

## Appendix A

### Test Results for the Standard and Tensor Methods

The columns in Tables A.1 - A.3 have the following meanings :

Function: name of the problem.

$n$ : dimension of the problem.

$x_0$: starting point (from Moré, Garbow and Hillstrom [1981]). 1, 10 and 100 stand for $x_0$, $10x_0$ and $100x_0$, respectively.

itns: number of iterations.

fcns: number of function evaluations (including the necessary function evaluations for finite difference gradients and Hessians).

$x_*$ : two methods converge to the same minimizer if and only if they have the same letter in this column.

The abbreviations OF, OL and NC stand for overflow, over iteration limit, and convergence to a nonminimizer, respectively. The iteration limit was 120.

**Table A.1 -- Test Results for the Standard and Tensor Methods on Nonsingular Problems**

| Function | $n$ | $x_0$ | Tensor ($p \geq 1$) | | | Tensor ($p = 1$) | | | Standard | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | itns | fcns | $x_*$ | itns | fcns | $x_*$ | itns | fcns | $x_*$ |
| Rosenbrock | 2 | 1 | 15 | 150 | a | 14 | 146 | a | 22 | 183 | a |
| | | 10 | 35 | 345 | a | 35 | 345 | a | 64 | 542 | a |
| | | 100 | 86 | 833 | a | 83 | 795 | a | OL | - | - |
| | 10 | 1 | 21 | 1724 | a | 21 | 1683 | a | 21 | 1615 | a |
| | | 10 | OF | - | - | OF | - | - | 73 | 5605 | a |
| | | 100 | OF | - | - | OL | - | - | OL | - | - |
| | 30 | 1 | 21 | 11640 | a | 25 | 13240 | a | 22 | 11610 | a |
| | | 10 | OF | - | - | 83 | 44067 | a | 92 | 48484 | a |
| | | 100 | OF | - | - | OL | - | - | OL | - | - |
| Wood | 4 | 1 | 30 | 629 | a | 30 | 629 | a | 62 | 1230 | a |
| | | 10 | 32 | 674 | a | 34 | 723 | a | 70 | 1391 | a |
| | | 100 | OL | - | - | OF | - | - | OL | - | - |
| Helical valley | 3 | 1 | 11 | 170 | a | 9 | 138 | a | 13 | 175 | a |
| | | 10 | 14 | 213 | a | 14 | 215 | a | 16 | 216 | a |
| | | 100 | 13 | 194 | a | 13 | 194 | a | 15 | 203 | a |
| Trigono-metric | 2 | 1 | 4 | 40 | a | 4 | 40 | a | 4 | 37 | a |
| | | 10 | 6 | 62 | a | 6 | 62 | a | 7 | 59 | a |
| | | 100 | 4 | 38 | a | 4 | 38 | a | 5 | 43 | a |
| | 10 | 1 | 6 | 553 | a | 7 | 555 | a | 19 | 1467 | a |
| | | 10 | 9 | 708 | a | 9 | 708 | a | 50 | 3820 | a |
| | | 100 | 40 | 3106 | a | 43 | 3336 | a | 32 | 2446 | a |
| Beale | 2 | 1 | 7 | 71 | a | 7 | 71 | a | 7 | 62 | a |
| | | 10 | 12 | 120 | a | 8 | 78 | a | OL | - | - |
| | | 100 | OL | - | - | OF | - | - | NC | - | - |
| Brown and Dennis | 4 | 1 | 13 | 272 | a | 13 | 271 | a | 18 | 347 | a |
| | | 10 | 16 | 333 | a | 14 | 292 | a | 24 | 461 | a |
| | | 100 | 22 | 469 | a | 21 | 440 | a | 40 | 778 | a |
| Brown badly scaled | 2 | 1 | OL | - | - | OF | - | - | OL | - | - |
| | | 10 | OL | - | - | OL | - | - | OL | - | - |
| | | 100 | OL | - | - | OL | - | - | OL | - | - |
| Box three dimensional | 3 | 1 | 12 | 183 | a | 13 | 200 | a | 21 | 290 | a |
| | | 10 | 19 | 305 | a | 23 | 359 | a | 59 | 827 | b |
| | | 100 | 20 | 325 | a | 17 | 281 | b | OL | - | - |

Table A.1 (continued)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Penalty I | 4 | 1 | 10 | 211 | a | 9 | 190 | b | 33 | 644 | c |
| | | 10 | 10 | 209 | a | 11 | 229 | a | 39 | 757 | a |
| | | 100 | 14 | 293 | a | 13 | 272 | a | 43 | 829 | a |
| | 10 | 1 | 15 | 1179 | a | 14 | 1101 | a | 35 | 2680 | a |
| | | 10 | 11 | 868 | a | 9 | 707 | a | 40 | 3059 | a |
| | | 100 | 23 | 1811 | a | 24 | 1889 | a | 52 | 3988 | a |
| | 30 | 1 | 19 | 10058 | a | 20 | 10577 | a | 36 | 18973 | a |
| | | 10 | 22 | 11640 | a | 26 | 13761 | a | 44 | 23189 | a |
| | | 100 | 29 | 15337 | a | 31 | 16391 | a | 99 | 52174 | a |
| Penalty II | 4 | 1 | 7 | 151 | a | 5 | 108 | a | 120 | 2285 | b |
| | | 10 | 13 | 274 | a | 20 | 416 | b | OL | - | - |
| | | 100 | 44 | 913 | a | 29 | 605 | b | OL | - | - |
| Variably dimensioned | 4 | 1 | 7 | 153 | a | 7 | 153 | a | 10 | 195 | a |
| | | 10 | 7 | 150 | a | 7 | 148 | a | 11 | 214 | a |
| | | 100 | 13 | 272 | a | 13 | 271 | a | 18 | 348 | a |
| | 10 | 1 | 10 | 794 | a | 11 | 863 | a | 16 | 1229 | a |
| | | 10 | 9 | 936 | a | 10 | 786 | a | 20 | 1534 | a |
| | | 100 | 18 | 1492 | a | 17 | 1338 | a | 32 | 2447 | a |
| | 30 | 1 | 18 | 9540 | a | 10 | 5306 | a | 33 | 17391 | a |
| | | 10 | 17 | 10056 | a | 17 | 9002 | a | 40 | 21073 | a |
| | | 100 | 56 | 30641 | a | OL | - | - | 112 | 58956 | a |
| Biggs EXP6 | 6 | 1 | 27 | 972 | a | 28 | 1005 | a | OL | - | - |
| | | 10 | 83 | 2987 | a | 34 | 1223 | b | OL | - | - |
| | | 100 | 25 | 914 | a | 26 | 937 | b | OL | - | - |
| Chebyquad | 6 | 1 | 6 | 221 | a | 6 | 221 | a | 14 | 484 | a |
| | | 10 | 34 | 1214 | a | 29 | 1033 | b | OL | - | - |
| | | 100 | 50 | 1785 | a | 53 | 1885 | b | OL | - | - |
| | 20 | 1 | 14 | 3815 | a | 16 | 4064 | a | OL | - | - |
| | | 10 | OF | - | - | 95 | 24102 | a | OL | - | - |
| | | 100 | OF | - | - | 104 | 26385 | a | NC | - | - |
| Watson | 6 | 1 | 11 | 403 | a | 11 | 397 | a | 19 | 658 | a |
| | 20 | 1 | OF | - | - | NC | - | - | OL | - | - |

**Table A.2 -- Test Results for the Standard and Tensor Methods on Singular (rank n-1) Problems**

| Function | $n$ | $x_0$ | Tensor ($p \geq 1$) | | | Tensor ($p=1$) | | | Standard | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | itns | fcns | $x_*$ | itns | fcns | $x_*$ | itns | fcns | $x_*$ |
| Rosenbrock | 2 | 1 | 31 | 305 | a | 31 | 305 | a | 70 | 570 | a |
| | | 10 | 39 | 383 | a | 39 | 383 | a | 93 | 764 | a |
| | | 100 | 60 | 601 | a | 58 | 579 | a | OL | - | - |
| | 10 | 1 | 11 | 945 | a | 15 | 1181 | b | 15 | 1162 | c |
| | | 10 | 25 | 2049 | a | 27 | 2125 | b | 27 | 2071 | c |
| | | 100 | 29 | 2272 | a | 41 | 3224 | b | 66 | 5060 | c |
| | 30 | 1 | 6 | 3202 | a | 6 | 3202 | a | 60 | 31598 | b |
| | | 10 | 12 | 6368 | a | 12 | 6367 | b | 23 | 12137 | b |
| | | 100 | 21 | 11647 | a | 21 | 11118 | a | 35 | 18453 | b |
| Wood | 4 | 1 | 25 | 519 | a | 25 | 519 | a | 37 | 708 | a |
| | | 10 | 34 | 713 | a | 36 | 758 | a | 52 | 1001 | a |
| | | 100 | 77 | 1576 | a | 58 | 1200 | a | OL | - | - |
| Helical valley | 3 | 1 | 9 | 147 | a | 9 | 138 | a | 13 | 180 | a |
| | | 10 | 15 | 229 | a | 15 | 222 | a | 25 | 334 | a |
| | | 100 | 20 | 319 | a | 16 | 243 | a | 26 | 345 | a |
| Trigono-metric | 2 | 1 | 6 | 59 | a | 6 | 59 | a | 12 | 102 | a |
| | | 10 | 5 | 50 | a | 5 | 50 | a | 9 | 75 | a |
| | | 100 | 6 | 58 | a | 6 | 58 | a | 9 | 76 | a |
| | 10 | 1 | 8 | 630 | a | 7 | 553 | a | 14 | 1085 | a |
| | | 10 | 14 | 1097 | a | 14 | 1098 | a | 15 | 1159 | a |
| | | 100 | 20 | 1639 | a | 18 | 1399 | a | NC | - | - |
| Beale | 2 | 1 | 6 | 64 | a | 7 | 74 | a | 6 | 52 | a |
| | | 10 | 10 | 100 | a | 10 | 100 | a | 47 | 396 | b |
| | | 100 | 64 | 679 | a | 23 | 231 | a | 44 | 364 | a |
| Brown and Dennis | 4 | 1 | 12 | 254 | a | 13 | 274 | a | 19 | 366 | a |
| | | 10 | 16 | 337 | a | 14 | 293 | a | 24 | 461 | a |
| | | 100 | 20 | 421 | a | 21 | 443 | a | 40 | 778 | a |
| Brown badly scaled | 2 | 1 | OF | - | - | OF | - | - | NC | - | - |
| | | 10 | OL | - | - | OF | - | - | NC | - | - |
| | | 100 | OL | - | - | OF | - | - | NC | - | - |
| Box three dimensional | 3 | 1 | 9 | 136 | a | 18 | 270 | b | 9 | 121 | c |
| | | 10 | 11 | 169 | a | 28 | 423 | b | 83 | 1083 | c |
| | | 100 | 28 | 430 | a | 25 | 380 | a | 31 | 412 | b |

Table A.2 (continued)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Penalty I | 4 | 1 | 4 | 84 | a | 4 | 84 | a | 15 | 290 | a |
| | | 10 | 4 | 84 | a | 4 | 84 | a | 20 | 385 | a |
| | | 100 | 7 | 147 | a | 7 | 147 | a | 26 | 499 | a |
| | 10 | 1 | 4 | 318 | a | 4 | 318 | a | 18 | 1381 | a |
| | | 10 | 7 | 550 | a | 7 | 550 | a | 24 | 1838 | a |
| | | 100 | 17 | 1341 | a | 17 | 1341 | a | 35 | 2685 | a |
| | 30 | 1 | 7 | 3722 | a | 10 | 5300 | a | 22 | 11605 | a |
| | | 10 | 17 | 9001 | a | 16 | 8471 | a | 29 | 15292 | a |
| | | 100 | 20 | 10584 | a | 18 | 9532 | a | 86 | 45334 | a |
| Penalty II | 4 | 1 | 6 | 130 | a | 4 | 88 | a | 57 | 1098 | b |
| | | 10 | 11 | 229 | a | 11 | 229 | a | 13 | 252 | b |
| | | 100 | 17 | 354 | a | 15 | 313 | a | OL | - | - |
| Variably dimensioned | 4 | 1 | 4 | 91 | a | 4 | 87 | a | 17 | 328 | a |
| | | 10 | 9 | 192 | a | 11 | 229 | a | 19 | 336 | a |
| | | 100 | 17 | 359 | a | 18 | 372 | a | 25 | 480 | a |
| | 10 | 1 | 11 | 1020 | a | 16 | 1254 | a | 24 | 1837 | a |
| | | 10 | 19 | 1479 | a | 17 | 1332 | a | 27 | 2066 | a |
| | | 100 | 20 | 1712 | a | 21 | 1641 | a | 40 | 3055 | a |
| | 30 | 1 | 48 | 25382 | a | 23 | 12170 | a | 41 | 21599 | a |
| | | 10 | OF | - | - | 36 | 19039 | a | OL | - | - |
| | | 100 | OF | - | - | OF | - | - | OL | - | - |
| Biggs EXP6 | 6 | 1 | 83 | 2962 | a | OF | - | - | OL | - | - |
| | | 10 | 74 | 2694 | a | 63 | 2246 | b | OL | - | - |
| | | 100 | OF | - | - | OF | - | - | OL | - | - |
| Chebyquad | 6 | 1 | 9 | 332 | a | 10 | 365 | a | 92 | 3135 | b |
| | | 10 | 22 | 789 | a | 19 | 683 | b | OL | - | - |
| | | 100 | 35 | 1245 | a | 22 | 785 | b | OL | - | - |
| | 20 | 1 | 29 | 7349 | a | 8 | 2045 | b | OL | - | - |
| | | 10 | 26 | 7349 | a | OF | - | - | OL | - | - |
| | | 100 | 50 | 13903 | a | 57 | 14441 | a | OL | - | - |
| Watson | 6 | 1 | 8 | 295 | a | 8 | 295 | a | 8 | 297 | a |
| | 20 | 1 | 24 | 6099 | a | 28 | 7100 | a | OL | - | - |

**Table A.3 -- Test Results for the Standard and Tensor Methods on Singular (rank n-2) Problems**

| Function | $n$ | $x_0$ | Tensor ($p \geq 1$) | | | Tensor ($p=1$) | | | Standard | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | itns | fcns | $x_*$ | itns | fcns | $x_*$ | itns | fcns | $x_*$ |
| Rosenbrock | 2 | 1 | 7 | 68 | a | 7 | 71 | a | 16 | 131 | b |
| | | 10 | 5 | 52 | a | 5 | 52 | a | 21 | 171 | b |
| | | 100 | 11 | 111 | a | 13 | 128 | b | 26 | 211 | c |
| | 10 | 1 | 11 | 872 | a | 11 | 872 | a | 14 | 1084 | b |
| | | 10 | 33 | 2718 | a | 25 | 1959 | b | 29 | 2233 | b |
| | | 100 | 25 | 2007 | a | 29 | 2330 | a | 45 | 3473 | a |
| | 30 | 1 | 11 | 5841 | a | 11 | 5841 | a | 21 | 11083 | a |
| | | 10 | 23 | 12172 | a | 25 | 1959 | a | OL | - | - |
| | | 100 | OF | - | - | 59 | 31247 | a | OL | - | - |
| Wood | 4 | 1 | 13 | 277 | a | 13 | 275 | a | 19 | 366 | b |
| | | 10 | 16 | 339 | a | 18 | 378 | b | 24 | 461 | c |
| | | 100 | OF | - | - | OF | - | - | NC | - | - |
| Helical valley | 3 | 1 | 15 | 222 | a | 13 | 196 | a | 41 | 541 | a |
| | | 10 | 21 | 316 | a | 19 | 281 | a | 49 | 648 | a |
| | | 100 | 22 | 328 | a | 21 | 315 | a | 47 | 621 | a |
| Trigono-metric | 2 | 1 | 4 | 38 | a | 4 | 38 | a | 8 | 67 | a |
| | | 10 | 6 | 62 | a | 6 | 62 | a | 10 | 83 | a |
| | | 100 | 7 | 70 | a | 7 | 70 | a | 8 | 67 | a |
| | 10 | 1 | 6 | 553 | a | 7 | 554 | a | 15 | 1161 | a |
| | | 10 | 11 | 941 | a | 11 | 864 | a | 15 | 1159 | a |
| | | 100 | NC | - | - | NC | - | - | NC | - | - |
| Beale | 2 | 1 | 6 | 65 | a | 6 | 65 | a | 10 | 84 | b |
| | | 10 | 10 | 101 | a | 9 | 95 | b | 10 | 84 | c |
| | | 100 | 22 | 235 | a | 20 | 217 | a | NC | - | - |
| Brown and Dennis | 4 | 1 | 12 | 253 | a | 13 | 271 | a | 18 | 347 | a |
| | | 10 | 17 | 355 | a | 14 | 292 | a | 24 | 461 | a |
| | | 100 | 20 | 424 | a | 20 | 419 | a | 40 | 778 | a |
| Brown badly scaled | 2 | 1 | NC | - | - | NC | - | - | NC | - | - |
| | | 10 | NC | - | - | NC | - | - | NC | - | - |
| | | 100 | NC | - | - | NC | - | - | NC | - | - |
| Box three dimensional | 3 | 1 | 16 | 248 | a | 11 | 172 | b | 16 | 200 | c |
| | | 10 | 22 | 331 | a | 13 | 202 | a | 22 | 304 | b |
| | | 100 | 48 | 763 | a | 40 | 633 | b | 46 | 637 | b |

Table A.3 (continued)

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Penalty I | 4 | 1 | 3 | 64 | a | 3 | 64 | a | 15 | 290 | a |
| | | 10 | 4 | 84 | a | 4 | 84 | a | 20 | 385 | a |
| | | 100 | 7 | 147 | a | 7 | 147 | a | 26 | 499 | a |
| | 10 | 1 | 4 | 318 | a | 4 | 318 | a | 18 | 1381 | a |
| | | 10 | 6 | 437 | a | 7 | 555 | a | 24 | 1838 | a |
| | | 100 | 19 | 1469 | a | 18 | 1418 | a | 35 | 2685 | a |
| | 30 | 1 | 9 | 4776 | a | 8 | 4246 | a | 22 | 11605 | a |
| | | 10 | 17 | 9002 | a | 15 | 7944 | a | 29 | 15292 | a |
| | | 100 | 16 | 8473 | a | 18 | 9525 | a | 86 | 45334 | a |
| Penalty II | 4 | 1 | 4 | 88 | a | 4 | 88 | a | 5 | 100 | a |
| | | 10 | 11 | 230 | a | 12 | 253 | a | 14 | 271 | b |
| | | 100 | 15 | 314 | a | 16 | 335 | b | 20 | 385 | c |
| Variably dimensioned | 4 | 1 | 9 | 192 | a | 9 | 192 | a | 13 | 254 | b |
| | | 10 | 9 | 193 | a | 9 | 191 | a | 42 | 804 | b |
| | | 100 | 18 | 377 | a | 12 | 250 | b | OL | - | - |
| | 10 | 1 | 21 | 1663 | a | 11 | 868 | b | 50 | 3818 | c |
| | | 10 | 16 | 1217 | a | 14 | 1100 | b | 102 | 7769 | c |
| | | 100 | 24 | 1966 | a | 21 | 1666 | b | OL | - | - |
| | 30 | 1 | 28 | 15368 | a | 16 | 8476 | b | 39 | 20547 | c |
| | | 10 | 41 | 23269 | a | OL | - | - | OL | - | - |
| | | 100 | 30 | 16426 | a | OL | - | - | OL | - | - |
| Biggs EXP6 | 6 | 1 | OF | - | - | OF | - | - | OL | - | - |
| | | 10 | 72 | 2627 | a | 54 | 1946 | b | OL | - | - |
| | | 100 | 41 | 1491 | a | 52 | 1884 | b | OL | - | - |
| Chebyquad | 6 | 1 | 13 | 473 | a | 11 | 404 | b | 96 | 3271 | c |
| | | 10 | 30 | 1076 | a | 26 | 926 | a | OL | - | - |
| | | 100 | 37 | 1320 | a | 32 | 1141 | a | OL | - | - |
| | 20 | 1 | 12 | 3308 | a | 13 | 3310 | a | OL | - | - |
| | | 10 | OF | - | - | 42 | 10631 | a | OL | - | - |
| | | 100 | 58 | 15705 | a | 46 | 11687 | a | OL | - | - |
| Watson | 6 | 1 | 6 | 216 | a | 6 | 216 | a | 6 | 211 | a |
| | 20 | 1 | OF | - | - | 33 | 8369 | a | OL | - | - |

Table A.3 (continued)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Powell | 4 | 1 | 9 | 194 | a | 11 | 233 | a | 15 | 290 | a |
| singular | | 10 | 12 | 253 | a | 14 | 294 | a | 21 | 404 | a |
| | | 100 | 22 | 464 | a | 22 | 464 | a | 26 | 499 | a |
| | 20 | 1 | 13 | 3308 | a | 10 | 2550 | a | 16 | 4043 | a |
| | | 10 | 11 | 3308 | a | 14 | 3559 | a | 21 | 5296 | a |
| | | 100 | 17 | 5075 | a | OF | - | - | 27 | 6801 | a |